## AMENDMENTS TO CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

**Listing of Claims:**

1. (Currently Amended) A method for protecting ~~the~~ a program run at the call of subprograms, ~~the~~ a called program performing, before or during the program execution, a check of ~~the~~ data passed directly or indirectly from ~~the~~ a calling program, characterized in that

- the calling program forms a first check sum for ~~the~~ parameters to be passed (step 2),
- said first check sum is stored in a specially provided memory area,
- the called program forms, before its execution, a second check sum for ~~the~~ received parameters (step 5) and checks it for equality with the first check sum (step 6), and
- in case of inequality of the first and second check sums the program is terminated (step 7) or an error message outputted.

2. (Currently Amended) A method for protecting ~~the~~ a program run at the call of subprograms, ~~the~~ a called program performing, before or during the program execution, a check of ~~the~~ data passed directly or indirectly from ~~the~~ a calling program, characterized in that upon call of a subprogram a timer is started (step 22) which counts ~~the~~ a number of clock cycles required for executing the called program and terminates the called program if the preset number of clock cycles was exceeded before termination of the ~~subprogram~~ called program (step 26).

3. (Original) A method according to claim 1, characterized in that the memory area for storing the check sum is a RAM or register area.

4. (Currently Amended). A method according to claim 1, characterized in that ~~the~~ a return ~~addresses~~ address of ~~the~~ a calling function ~~are~~ is entered in a table and the called program checks the return address reported by the calling program (step 13) by checking the presence of said return address on the basis of the table.

5. (Currently Amended). A method according to claim 2, characterized in that ~~the~~ a timer value is read at certain preset points (step 24) and compared with a likewise preset intermediate value (step 25) and the called program is terminated if the preset intermediate value was exceeded (step 26).